Alma Sistemi SRL
consulting for space

Economy bY spacE (EYE)

Marie Skłodowska-Curie Actions (MSCA)
Research and Innovation Staff Exchange (RISE)
H2020-MSCA-RISE-2020
G.A. 101007638

Ship Detection in Souda, Chania (SAR-data) using spatial variations in illumination

Stavrakakis Konstantinos

24/01/2022

This project is co-funded by
the European Union

Setup & Tools:

- Anaconda Environment
- Gpt (SNAP) Installed
- PEPS Credentials (for data download)
- Python Libraries: rasterio, folium, imageio, eodag

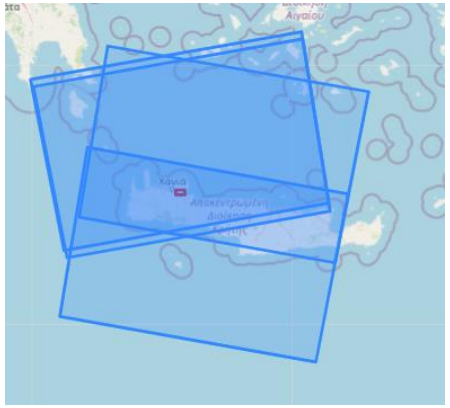
Processing workflow

Step 1
Request & Choose Data of Interest:

Code:

```
product_type = 'S1_SAR_GRD'
extent = {
    'lonmin': 24.044031,
    'lonmax': 24.143720,
    'latmin': 35.462626,
    'latmax': 35.506875
}
products, estimated_nbr_of_results = dag.search(
    productType=product_type,
    start='2017-06-01',
    end='2017-09-02',
    geom=extent
)
```

Tiles on Map:



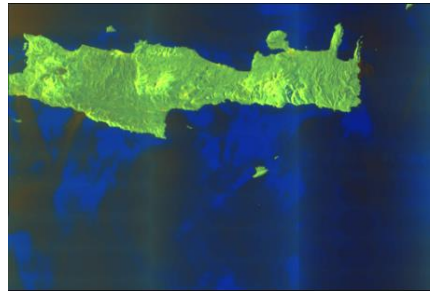
Step 2:
Download tile of Interest and Preview

```
product_path = product.download()
from IPython.display import display, Image
loc1 = 'C:\Programming\Snap\Chania_SD\S1A_IW_GRDH_1SDV_20180501T042412_20180501T042437_021706_025716_20C9.SAFE'

product_path = loc1


product_path = product_path[7:] if product_path.startswith('file://') else product_path

prod = product_path
#product_path = prod
Image(os.path.join(prod, 'preview/quick-look.png'))
```



Step 3

Subset & Orbit File Application to downloaded tile:

```
graph_subset = os.path.join(workspace, 'Subset.xml')
with open(graph_subset, 'w') as g_1:
    g_1.write(
"""
<graph id="Graph">
  <version>1.0</version>
    <node id="Read">
      <operator>Read</operator>
      <sources/>
      <parameters>
        <file>${inputproduct}</file>
      </parameters>
    </node>
    <node id="Subset">
      <operator>Subset</operator>
      <sources>
        <sourceProduct refid="Read"/>
      </sources>
      <parameters>
        <region>15225,1925,17150,3150</region>
        <copyMetadata>true</copyMetadata>
      </parameters>
    </node>
    <node id="Apply-Orbit-File">
      <operator>Apply-Orbit-File</operator>
      <sources>
        <sourceProduct refid="Subset"/>
      </sources>
    </node>
    <node id="Write">
      <operator>Write</operator>
      <sources>
        <sourceProduct refid="Apply-Orbit-File"/>
      </sources>
      <parameters>
        <file>${outputproduct}</file>
      </parameters>
    </node>
</graph>
"""
)
os.environ['LD_LIBRARY_PATH'] = '.'
!gpt {graph_subset} -Pinputproduct={product_path} \
-Poutputproduct={os.path.join(workspace, 'S1A_IW_GRDH_1SDV_20180501T042412_20C9_Orb_new')}
```

Step 4

Ship Detection Algorithm:

1. Mask the land using default settings.
2. Calibrate VH,VV and choose SigmaBand
3. AdaptiveThresholding (To find objects in the ocean)
   * smallest ship is 30m long
   * largest ship is 500m long
4. ObjectDiscrimination (Default parameters 30 and 600

```
graph_process = os.path.join(workspace, 'ShipDetection.xml')
with open(graph_process, 'w') as g_2:
    g_2.write(
"""
<graph id="Graph">
  <version>1.0</version>
    <node id="Read">
      <operator>Read</operator>
      <sources/>
      <parameters>
        <file>${inputproduct}</file>
      </parameters>
    </node>
    <node id="Land-Sea-Mask">
      <operator>Land-Sea-Mask</operator>
      <sources>
        <sourceProduct refid="Read"/>
      </sources>
      <parameters>
        <landMask>true</landMask>
        <useSRTM>true</useSRTM>
        <invertGeometry>false</invertGeometry>
        <shorelineExtension>10</shorelineExtension>
      </parameters>
    </node>
    <node id="Calibration">
      <operator>Calibration</operator>
      <sources>
        <sourceProduct refid="Land-Sea-Mask"/>
      </sources>
      <parameters>
        <selectedPolarisations>VH,VV</selectedPolarisations>
        <outputSigmaBand>true</outputSigmaBand>
      </parameters>
    </node>
    <node id="AdaptiveThresholding">
      <operator>AdaptiveThresholding</operator>
      <sources>
        <sourceProduct refid="Calibration"/>
      </sources>
      <parameters>
        <targetWindowSizeInMeter>30</targetWindowSizeInMeter>
        <guardWindowSizeInMeter>500.0</guardWindowSizeInMeter>
        <backgroundWindowSizeInMeter>800.0</backgroundWindowSizeInMeter>
        <pfa>12.5</pfa>
      </parameters>
    </node>
    <node id="Object-Discrimination">
      <operator>Object-Discrimination</operator>
      <sources>
        <sourceProduct refid="AdaptiveThresholding"/>
      </sources>
      <parameters>
        <minTargetSizeInMeter>30</minTargetSizeInMeter>
        <maxTargetSizeInMeter>600</maxTargetSizeInMeter>
      </parameters>
    </node>
    <node id="Write1">
      <operator>Write</operator>
      <sources>
        <sourceProduct refid="Object-Discrimination"/>
      </sources>
      <parameters>
        <file>${outputproduct1}</file>
      </parameters>
    </node>
    <node id="Write2">
      <operator>Write</operator>
      <sources>
        <sourceProduct refid="Object-Discrimination"/>
      </sources>
      <parameters>
        <file>${outputproduct2}</file>
        <formatName>Geotiff</formatName>
      </parameters>
    </node>
</graph>
"""
)
!gpt {graph_process} -Pinputproduct={os.path.join(workspace, 'subset_0_of_S1A_IW_GRDH_1SDV_20180501T042412_20180501T042437_021706_025716_20C9_Orb.dim')} \
-Poutputproduct1={os.path.join(outPos, 'S1A_IW_GRDH_1SDV_20180501T042412_20C9_processed')} -Poutputproduct2={os.path.join(outPos, 'S1A_IW_GRDH_1SDV_20180501T042412_20C9_processed')}
```

Step 5

Visualize the Output:

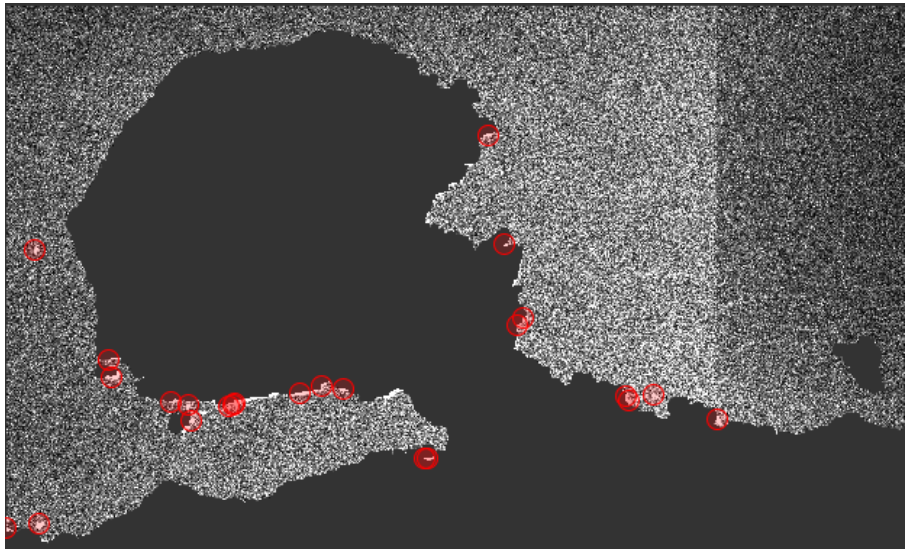
1. Subset the processed file
2. Apply Terrain Correction

```
graph_visu = os.path.join(workspace, 'vizualisation_ships.xml')
with open(graph_visu, 'w') as g_3:
    g_3.write(
"""
<graph id="Graph">
  <version>1.0</version>
    <node id="Read">
      <operator>Read</operator>
      <sources/>
      <parameters>
        <file>${inputproduct}</file>
      </parameters>
    </node>
    <node id="SubSet">
      <operator>Subset</operator>
      <sources>
        <sourceProduct refid="Read"/>
      </sources>
      <parameters>
        <region>0,1000,10000,10000</region>
      </parameters>
    </node>
    <node id="Terrain-Correction">
      <operator>Terrain-Correction</operator>
      <sources>
        <sourceProduct refid="SubSet"/>
      </sources>
      <parameters>
        <sourceBands>Sigma0_VH_ship_bit_msk</sourceBands>
        <nodataValueAtSea>False</nodataValueAtSea>
      </parameters>
    </node>
    <node id="Write">
      <operator>Write</operator>
      <sources>
        <sourceProduct refid="Terrain-Correction"/>
      </sources>
      <parameters>
        <file>${outputproduct}</file>
        <formatName>Geotiff</formatName>
      </parameters>
    </node>
</graph>
"""
)
!gpt {graph_visu} -Pinputproduct={os.path.join(outPos, 'S1A_IW_GRDH_1SDV_20180501T042412_20C9_processed.dim')} \
-Poutputproduct={os.path.join(outPos, 'subset_visualization')}
```

Results

Raw result of Ocean Object Detection (Ship)

(prior terrain correction)



Result of Ocean Object Detection (Ship) on interactive map

(post terrain correction)

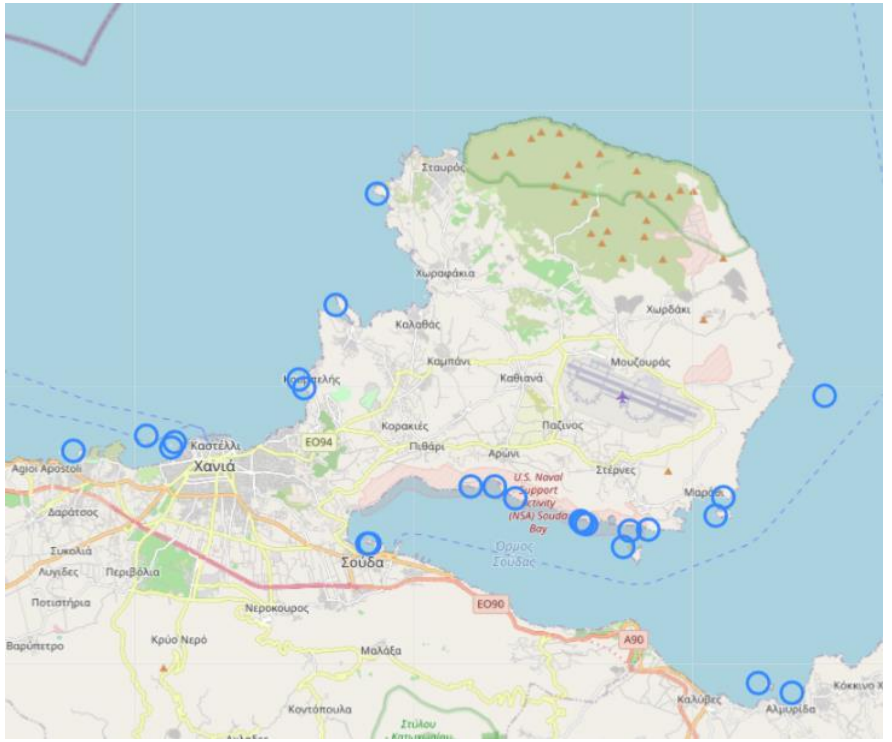


Image Notes:

- Land has dark-gray color. (*masked)*
- Ocean has light-gray color
- Red circles are the detected objects interpreted as ships

Code Available

https://github.com/razkey23/Ship-Detection-SAR/blob/main/ShipDetectionChania.ipynb